Sales transaction system using redis database

Ankitkumar Patel, Jeongkyu Lee (Advisor)
Department of Computer Science & Engineering
University of Bridgeport, Bridgeport, CT

Abstract

A traditional Client Server system provides poor performance on read and write process in terms of throughput and latency because all servers use their own memory to handle the entire process, which is quite time consuming. In order to address this, we employ NoSQL database, specifically Redis. Redis is open-source in-memory database and one of the most popular NoSQL database for key-value data. Redis is used to store advanced key-value data in-memory on distributed cache. In distributed cache, all servers are connected to a cloud resource, and use that cache memory to store and retrieve data frequently. With the helps of Redis, the application for sales transactions perform read and write operations much faster.

Introduction

A company delivers products to its customers worldwide and keep track of records of sales transaction for that it has to maintain records of customers and products which are purchased by customers. It checks how many customers have purchased products and when they did the transaction. Sales transaction details manage date of transaction, product, and price of products, longitude and latitude.

Implementation of this web application is developed in ASP.NET with c# as code behind the model to provide API while redis-client and redis-server run in backend to deal with data. To check the efficiency of the application, a CSV (Comma separated value) file with big data is uploaded into the system. Data file contains millions of records of sales transaction and to store the data using redis in-memory cache with the use of advanced data types such as strings, hashes, sets, sorted sets. Redis is the advanced key-value database generates unique key for each rows which consider as value or single item like transaction date, price columns can also be stored by assigning a key to that column value.
.

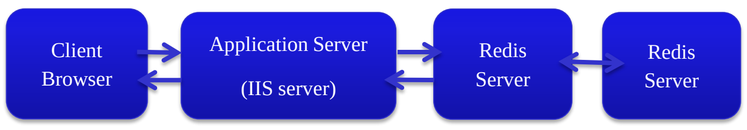
Why redis database?

Distributed Cache: Distributed cache means the data is stored on other cloud resources rather than on individual web server's memory, this is what exactly with applications with relational database. In distributed cache scenario, data is available to all application's web servers. Redis provides distributed cache environment by storing the data in memory. In order to offer best performance, redis sync cache from all the web servers.

Data Structure: Redis offers very effective and efficient data structure at a level that it is also called as a data structure server. The key stored in a database can be hashes, lists, and strings, sorted or unsorted sets.

Very fast: It considers as one of the fastest NOSQL server as it works well with in-memory datasets.

Client-Server Architecture

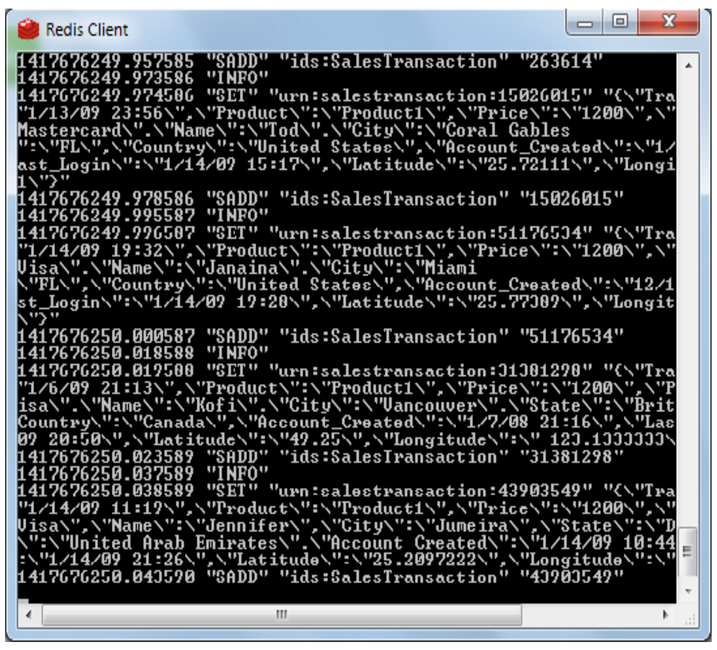


Client sends a request to application server, application server forward that request to redis client which is operate by redis server as a backend. If consider the sales transaction scenario using this architecture. It would be like when user tries to upload CSV file using browser. Browser sends a request to application server which forwards that request to redis client to perform operations to redis database by redis server. Redis server reply back to application server which forwards the same response to user's browser.

Methods to store data

RedisClient.Store() method is used for mass insertion of data.
RedisClient.Set(Key,Value) method is used to store single key and value data in Redis distribution cache.
RedisClient.Get(Key) method is used to get value of the key given by the user from redis.

Result



Here, the data stored in the database are in JSON script format. Redis will automatically assign a key to every data rows and the data rows are treated like a value of a unique key.

Conclusion

Redis is the solution for applications do input output operations on frequent basis because it works as a cache and the data structure is also simple and efficient for large data to get high throughput, low latency, high scalability and availability by using master-slave architecture.

References

- To get the redis libraries to work with redis database with Microsoft platform ASP.NET with C#. **https://servicestack.net/redis**
- Redis database commands and information in depth. **http://redis.io/**

Acknowledgement

My sincere thanks to my advisor Prof. Jeongkyu Lee from University of Bridgeport for excellent guidance, valuable time and suggestions to accomplish this project successfully.